

DRAWINGS:

3.1 Relational Database of the AC-knowledge:

3.1.1 ER-Diagram of the AC-Database:

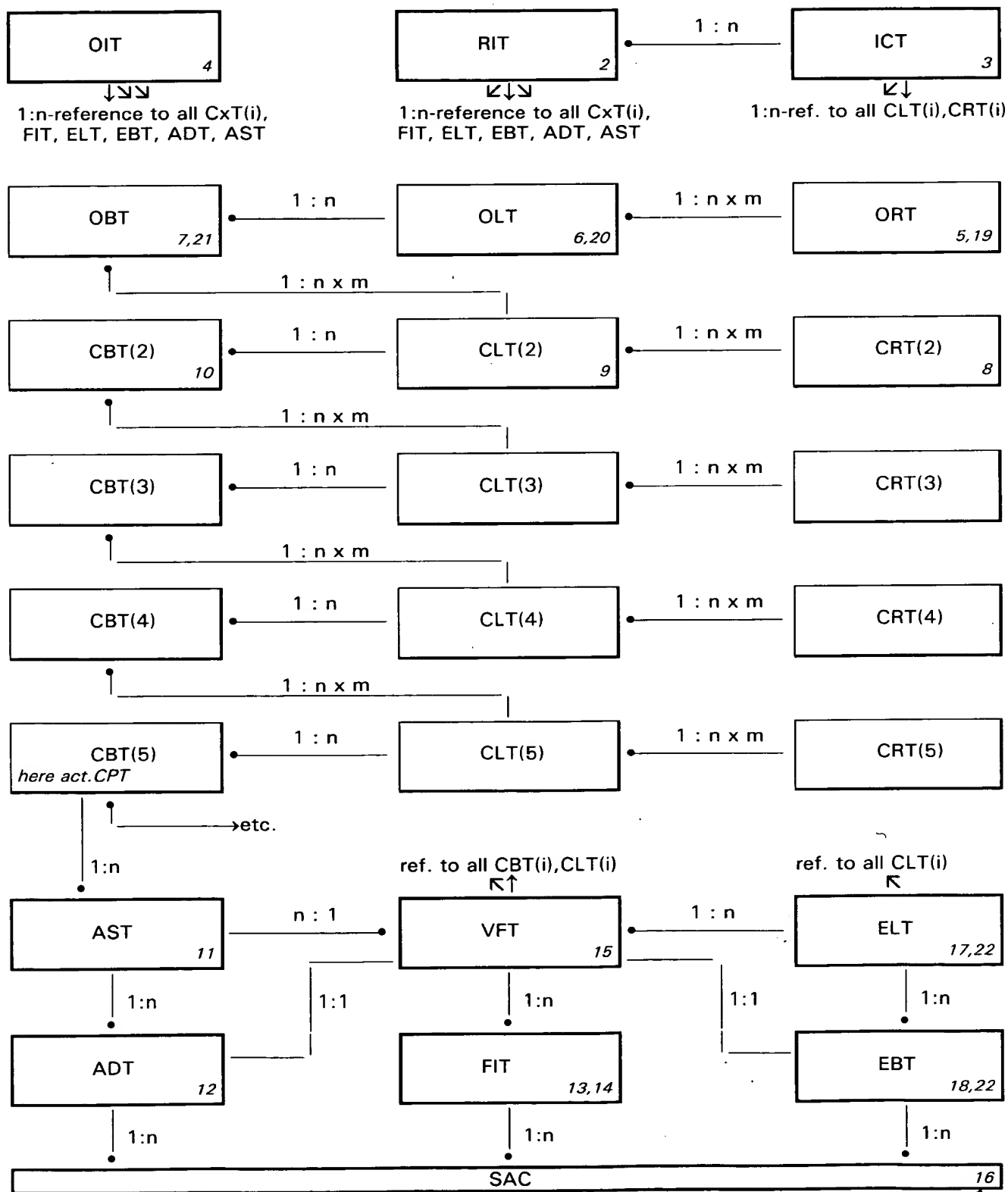


Fig. 1

<i>column:</i>	<i>datatype</i>	<i>value-range</i>	<i>meaning:</i>
----------------	-----------------	--------------------	-----------------

Fig. 2a

Fig.2b (RIT in a Motorola-Example)

<i>column:</i>	<i>datatype</i>	<i>value-range</i>	<i>meaning:</i>
----------------	-----------------	--------------------	-----------------

Fig. 3a

[no 2 equal register(-destination)-values]

Fig. 3b

Operation-Identification-Table: [OIT: every processor-operation gets an Oper.ID and an Oper.BitCode]
 column: datatype value-range meaning:

Operation ID (PK)	signed byte	-1..63	bit of the Calculation BitCode - see table below.
Operation BitCode (FK)	number	0..2 ¹²⁸ -1	2 ⁿ Calculation ID - see table below.
Operation Type	char(5)	5 Bytes	5 characters-code of the operation-type, see Fig. 4c
Operation Mnemonic	char(5)	5 Bytes	abbreviation of the operation - see table below.
Operation Description	varchar2(32)	≤32 Bytes	optional description of the operation - see table below

Fig. 4a

Operation ID	Operation BitCode	Operation Type	Op.Mnemonic	Operation Description
-1	0	???	unknown operation
0	1	.I11?	TST	set flags in dependence of reg.(-ref.)
1	2	.I12!	NEG	negation amount
2	4	.I12!	NOT	bitwise inversion
3	8	:I02	MOVI	const.integer→register(-reference)
4	16	:I12+	ADDI	add constant integer
5	32	:I12-	SUBI	subtract constant integer
6	64	:I13*	MULI	multiply constant integer
7	128	:I23/	DIVI	divide by constant integer
8	256	:I13%	MODI	rest of integer-division
9	512	:I12*	SHLI	integer-times a duplication
10	1.024	:I12/	SHRI	integer-times a halvation
11	2.048	:I12	ORI	set bits set in a constant integer
12	4.096	:I12&	ANDI	clear bits not set in a const. integer
13	8.192	:I12?	BTSTI	check if int-th bit is set in reg.(-ref.)
14	16.384	:I12?	CMPI	reg.(-ref.)-comparison with integer
15	32.768	II22	MOV	move src.-reg.(ref.)→dest.reg(ref.)
16	65.536	II22+	ADD	addition of register(-reference)
17	131.072	II22-	SUB	subtraction of register(-reference)
18	262.144	II23*	MUL	multiplication of register(-reference)
19	524.288	II33/	DIV	division by register(-reference)
20	1.048.576	II33%	MOD	rest of division by register(-ref.)
21	2.097.152	II22*	SHL	register(-ref.)-times a duplication
22	4.194.304	II22/	SHR	register(-ref.)-times a halvation
23	8.388.608	II22	OR	set bits set in of register(-reference)
24	16.777.216	II22&	AND	clear bits not set in register(-ref.)
25	33.554.432	II21?	BTST	check if reg.(-ref.)-th bit is set
26	67.108.864	II21?	CMP	compare reg.(-ref.)1 with reg.(-ref.)2
27	134.217.728	:P00.	JMP	add integer to PC _μ /EIP _π (=jump to)
28	268.435.456	CP1.<	JLT	jump if CMP<
29	536.870.912	CP1!>	JLE	jump if CMP≤
30	1.073.741.824	CP1.=	JEQ	jump if CMP=
31	2.147.483.648	CP1!<	JGE	jump if CMP≥
32	4.294.967.296	CP1!=	JNE	jump if CMP≠
33	\$2.0000.0000	CP1.>	JGT	jump if CMP>
34	\$4.0000.0000	CP1!<	JPL	jump if ≥ 0
35	\$8.0000.0000	CP1.<	JMI	jump if < 0
36	\$10.0000.0000	CP1.^	JCS	jump if carry-flag is set
37	\$20.0000.0000	CP1!^	JCC	jump if carry-flag is clear
38	\$40.0000.0000	CP1.~	JVS	jump if overflow is set
39	\$80.0000.0000	CP1!~	JVC	jump if overflow is clear
40	\$100.0000.0000	CP2.<	DJMP	decrement and jump if reg.(-ref.)<0
41	\$200.0000.0000	PS1..	CALL	PC _μ /EIP _π →-(USP _μ /ESP _π); +JUMP
42	\$400.0000.0000	SP11.	RET	(USP _μ /ESP _π)+→PC _μ /EIP _π
43	\$800.0000.0000	.I...	I???	unknown integer-operation
44	\$1000.0000.0000	.F...	F???	unknown floating-point-operation

09704803 "110300

OpCode-Register-Table: [ORT - by opcode icl, initial-conditions concerned registers and the effect]

column:	datatype	value-range	meaning:
---------	----------	-------------	----------

column:	datatype	value range	meaning:
OpCode (PK)	integer	$0..2^{32}-1$	complete instruction, truncated if > 4 bytes
IniConNr (PK)	signed byte	-31..30	current number of the used initial conditions
Register ID dest (PK)	signed byte	0..127	one by execution concerned destination-reg. (see RIT)
Register ID source (PK)	signed byte	-1,0..127	-1 or one possible source-register (see RIT).
value before change	integer	$0..2^{32}-1$	register(-reference)-value before it was changed
value after change	integer	$0..2^{32}-1$	register(-reference)-value after changing
gradient if unsigned	signed byte	-128..127	before/after-gradient, when defined as unsigned
gradient if signed	signed byte	-128..127	before/after-gradient, when defined as signed
value source	integer	$0..2^{32}-1$	value of a possible source-register(-reference)
Operations_BitCode	number	$0..2^{128}-1$	bitmask, which flags all possible operations between this Register_ID_dest / Register_ID_source-combination (p.e. $2 + 2 = 2 * 2$ using same reg.'s). Values see CIT, calculation see fig.19.

Fig.5

For every register- or register-reference-modification of the same opcode-execution one entry is generated, which gives information about the register-(reference)-values before and after the execution and further information about the degree of changing and with it a hint to a possible operation and a possible source-register which were used. (Packed-, nibble-, or BCD-operations are not considered.)

The last address-register is the stack-pointer. The last data-register is the "energy"-register.

An address-register can be every register which value can be a pointer to memory which destination can be accessed using this register as a reference.

Several registers can be modified simultaneous - therefore this additional 1:n table, where *Register_ID_dest* means the identity of the changed register. Sometimes many register(-references) could be the source for one operation - this quantum increases through the sum over all possible operations.

Therefore the following tables identify the used opcode and the concerned register(s):

[illegible]

column:		datatype	value-range	meaning:
OpCode	(PK)	integer	0..2 ³² -1	complete instruction, truncated if > 4 bytes
IniConNr	(PK)	signed byte	-31..30	number of used initial condition
active ChkSum corrupt		boolean	1 0	flag: checksum of active AC-program changed
inactive ChkSum corrupt		boolean	1 0	flag: checksum of inactive AC-program changed
Exception Vect changed		signed byte	-128...0	Register ID of the (first) overwritten exception-vector
multiple Exc Vect chg		boolean	1 0	more than one exception-vector was overwritten
Processor Mode Changed		boolean	1 0	flag: processor-mode changed (p.e. trace cleared)
Number of Exception		byte	0..N+1	exception-number [0: = no exception] (if 0 = exc.:+1)
OpCode_length_or_jump		signed byte	-128..127	EIP _π /PC _u after execution – EIP _π /PC _u before execution -128=\$FF=long back-jump, 127=\$7F=long forward j.
CCR before execution		byte	0..255	CC-flags, which could cause a jump.
Register_changed_BitCode		number	0..2 ¹²⁸ -1	$\exists 2^{\text{ORT}}.\text{Register ID dest} \forall \text{ORT}(\text{opcode}, \text{IniConNr})$
Register_source_BitCode		number	0..2 ¹²⁸ -1	$\exists 2^{\text{ORT}}.\text{Register ID source} \forall \text{ORT}(\text{opcode}, \text{IniConNr})$
max_Operations_BitCode		number(19)	0..2 ¹²⁸ -1	$\exists \text{ORT}.\text{Calculation BitCode} \forall \text{ORT}(\text{opcode}, \text{IniConNr})$
min_Operations_BitCode		number(19)	0..2 ¹²⁸ -1	$\zeta \text{ORT}.\text{Calculation BitCode} \forall \text{ORT}(\text{opcode}, \text{IniConNr})$
time of execution		integer	0..2 ³² -1	deci-seconds after {20.9.1994, 0:00:00, 0 o'clock}
cycles of execution		byte	1..255	clock-cycles the opcode-execution needed
aim valuation		signed byte	-128..127	aim-attaining-valuation using above initial-conditions
gradient aim valuation		signed byte	-128..127	-"-difference to $CLT(n-1, \text{IniConNr}).\text{aim valuation}$

Fig. 6

The following **combination-tables** are created dynamically - they have the same non-PK-columns, like the OBT respectively OLT respectively ORT, but for every additional number of combinations a further more opcode in the PK:

Combination-Register-Table: [CRT(i), i = number of opcodes in the combination, CRT(1) = ORT]

column:		datatype	value-range	meaning:
OpCode 1 (PK)		integer	0-2 ³² -1	opcode #1 (first of the combination)
{for all OpCodes} (PK)		all integer	0-2 ³² -1	{for all opcodes from #2 up to N-1}
OpCode N (PK)		integer	0-2 ³² -1	opcode#N (last of the combination)
IniConNr (PK)		signed byte	-31..30	current number of the used initial conditions
Register ID dest (PK)		signed byte	0..127	one by execution concerned destination-reg. (see RIT)
Register ID source (PK)		signed byte	-1..127	-1 or one possible source-register (see RIT).
{same non-PK columns like in the Opcode-Register-Table.}		see above	see above	same non-PK columns like ORT.

Fig.8

Combinations-Learn-Table: [CLT(i), i = number of opcodes in the combination, CLT(1) = OLT]

column:		datatype	value-range	meaning:
OpCode 1 (PK)		integer	0-2 ³² -1	opcode #1 (first of the combination)
{for all OpCodes} (PK)		all integer	0-2 ³² -1	{for all opcodes from #2 up to N-1}
OpCode N (PK)		integer	0-2 ³² -1	opcode#N (last of the combination)
IniConNr (PK)		signed byte	-31..30	current number of the used initial conditions
{same non-PK columns like in the Opcode-Learn-Table.}		see above	see above	same non-PK columns like OLT.

Fig.9

Combinations-Base-Table: [CBT(i), i = number of opcodes in the combination, CBT(1) = OBT]

column:		datatype	value-range	meaning:
OpCode 1 (PK)		integer	0-2 ³² -1	opcode #1 (first of the combination)
{for all OpCodes} (PK)		all integer	0-2 ³² -1	{for all opcodes from #2 up to N-1}
OpCode N (PK)		integer	0-2 ³² -1	opcode#N (last of the combination)
{same non-PK columns like in the Opcode-Base-Table.}		see above	see above	same non-PK columns like OBT.

Fig.10

CBT(max.) = CPT = Combination-Plan-Table = point of origin of the outcoming program.

09704803 "10300

Programming-aim and valuation-function tables:***Aim-Solution-Table: [AST - solutions of all programming-aims]***

column: datatype value-range meaning:

Aim ID (PK)	short	0..65535	Identifier of the programming-aim
Solution Nr (PK)	byte	0..255	number of the solution-program
aim_Program	long	String	opcode-combination of the solution here as a string
Program length	short	1..65535	length of the solution-program in doublewords
cycles of execution	integer	1..2 ³² -1	execution-time in clock-cycles of the solution-program
used Registers BitCode	number	1..2 ¹²⁸ -1	bitcode of all in the solution-program used registers
used Operations Bitcode	number	1..2 ¹²⁸ -1	bitcode of all in the solution-program used opcodes
used aim Valuation Func	signed short	0..32767	identifier of the used aim-distance valuation-function

Fig. 11***Aim-Description-Table: [ADT - identification and description of programming-aim]***

column: datatype value-range meaning:

Aim ID (PK)	short	0..65535	Identifier of the programming-aim
aim_Description	varchar2(32)	≤32 Bytes	description of the programming-aim
used Processor Mode	integer	0-2 ³² -1	flags above CC control-register-bits
all_dest Register BitCode	number	1..2 ¹²⁸ -1	bitcode of all output-registers in this task
all_source Register BitCode	number	1..2 ¹²⁸ -1	bitcode of all input-registers in this task
unused Register BitCode	number	1..2 ¹²⁸ -1	bitcode of all registers which should not be used
unused_Operation_BitCode	number	0..2 ¹²⁸ -1	bitcode of all opcode-IDs which are not allowed to use in this task (default = \$0000.0000:0000.0000)
aim_implement_solutions	long	String	string of the Aim_ID's (words) of earlier solutions, which could be implemented here.
aim_fulfill valuation mode	boolean	0 1	mode of aim-valuation: 0 = SQL ; 1 = machine-code
aim_fulfilled Flag Function	varchar2(99)	≤99 Bytes	boolean aim-attained recognition-function as a string
aim_Valuation FunctionID	signed short	0..32767	identifier of the valuation-function (see VFT)

Fig. 12***Functions-Identification-Table: [FIT - table of the basic subfunctions used in the valuation-function]***

a.) for SQL-functions:

column: datatype value-range meaning:

Function ID (PK)	signed byte	-1..127	identification-number of the basic function
Function BitCode	number(19)	0..2 ⁶⁴ -1	bitcode of this basic function (only one bit is set)
Function Name	char(5)	5 Bytes	function-name
Function Type	byte	0..99	0 = value, 1 = unitary, 2 = binary, 3 = ternary, ...
Function Flatten	signed byte	-127..127	degree of flattening [+ = steepening, - = flattening]
Function_Template	varchar2(99)	≤99 Bytes	SQL function-template
Function_Description	varchar2(99)	≤99 Bytes	optional description of the basic sub-function

Fig. 13a

09704303 " 110300

F.ID	Function BitCode	F.Name	F.T.	F.F.	Function Template	Function Description
0	1	NUM	0	0	< following value >	a constant number follows
1	2	ENGY	0	0	ELT.energy after	energy after execution
2	4	GRAD	0	0	ELT.energy_after -ELT.energy before	energy-gradient
3	8	VALUE	0	0	CLT(n). < columnNr >	value in the following column-number (last row)
4	16	EREG	0	0	< EnergyRegister_ID >	ID of the energy-register
5	32	SGN	1	0	SIGN(%s)	algebraic sign
6	64	ROUND	1	0	ROUND(%s, 0)	rounded
7	128	INT	1	0	FLOOR(%s)	truncated after dec.point
8	256	ABS	1	0	ABS(%s)	amount
9	512	NEG	1	0	-(%s)	negation
10	1.024	ADD	2	1	((%s) + (%s))	addition
11	2.048	SUB	2	-1	((%s) - (%s))	subtraction
12	4.096	MUL	2	4	((%s) * (%s))	multiplication
13	8.192	DIV	2	-4	((%s) / (%s))	division
14	16.384	MOD	2	-2	MOD(%s, %s)	rest of division
15	32.767	SQRT	1	-8	SQRT(%s)	square-root
16	\$1.0000	CBRT	1	-12	POWER(%s, 1/3)	cube-root
17	\$2.0000	MIN	2	-10	LEAST(%s, %s)	minimum
18	\$4.0000	MAX	2	-10	GREATEST(%s, %s)	maximum
19	\$8.0000	LN	1	-48	LN(%s)	natural logarithm
20	\$10.0000	EXP	1	48	EXP(%s)	nat. exponential-function
21	\$20.0000	LD	1	-32	LOG(2, %s)	logarithm on base 2
22	\$40.0000	POT2	1	32	POWER(2, %s)	2nd power of ...
23	\$80.0000	SIN	1	-64	SIN(%s)	sine
24	\$100.0000	COS	1	-64	COS(%s)	cosine
25	\$200.0000	TAN	1	127	TAN(%s)	tangent
26	\$400.0000	ASIN	1	127	ASIN(%s)	arc sine
27	\$800.0000	ACOS	1	127	ACOS(%s)	arc cosine
28	\$1000.0000	ATAN	1	-127	ATAN(%s)	arc tangent
29	\$2000.0000	SINH	1	40	SINH(%s)	sine hyperbolic
30	\$4000.0000	COSH	1	50	COSH(%s)	cosine hyperbolic
31	\$8000.0000	TANH	1	-127	TANH(%s)	tangent hyperbolic
32	\$1.0000.0000	LOG	2	-64	LOG(%s, %s)	logarithm
33	\$2.0000.0000	POT	2	64	POWER(%s, %s)	n-th power of ...
34	\$4.0000.0000	OR	2	1	((%s) (%s))	bitwise OR
35	\$8.0000.0000	AND	2	-1	((%s) & (%s))	bitwise AND
36	\$10.0000.0000	EQ	2	-127	DECODE(%s, %s, 1, 0)	equal
37	\$20.0000.0000	LE	2	-127	DECODE(GREATEST(%s - %s, 0), 0, 1, 0)	less-equal
38	\$40.0000.0000	GE	2	-127	DECODE(LEAST(%s - %s, 0), 0, 1, 0)	greater-equal
39	\$80.0000.0000	FRAME	1	-10	GREATEST(LEAST(%s, +127), -128)	frame to signed-byte: max. = 127, min. = -128
40	\$100.0000.0000	BITS	1	-64	(1 & %s) + (2 & %s) / 2 + (4 & %s) / 4 + (8 & %s) / 8 +	number of bits in the integer value
41	\$200.0000.0000	S_REG	0	0	ADT.all_source_Registers- BitCode	bitcode of the source- register
42	\$400.0000.0000	D_REG	0	0	ADT.all_dest_Registers BitCode	bitcode of dest.-register
43	\$800.0000.0000	AIM_F	0	0	VAL(ADT.aim_fulfilled_Flag- Function)	result of the boolean aim-fulfilled-function
...

Fig. 13b

09704803 110300

b.) for machine-code functions:

column: datatype value-range meaning:

Function ID (PK)	signed byte	-1..127	identification-number of the basic function
Function BitCode	number(19)	0..2 ⁶⁴ -1	bitcode of this sub-function
Operations BitCode	number	0..2 ¹²⁸ -1	bitcode of the used opcodes in this function
Registers BitCode	number	0..2 ¹²⁸ -1	bitcode of the used registers in this function
Function Name	char(5)	5 Bytes	short notation of this sub-function
Function Type	byte	0..99	0 = value, 1 = unitary, 2 = binary, 3 = ternary, ...
Function Flatten	signed byte	-128..127	degree of function-flattening ($1 \triangleq f(x) = x$)
Function OpCodes	number	1..2 ¹²⁸ -1	sub-function in machine-code
Function Description	varchar2(99)	≤99 Bytes	optional description of the sub-function

Fig. 14a

Func.ID	Func.BitCode	Oper.BitCode	Reg.BitCode	Func.Name	F.T.	Func.OpCodes	Function Descript.
0	1	\$A000.4008	<energy>	FRAME	1	s.b. Func.1	prevent overflow
1	2	\$28800.0009	<energy>	SGN	1	s.b. Func.2	signum
2	4	\$0000.0002	<energy>	NEG	1	<NEG>	negation
3	8	\$0000.0200	<energy>	MUL2	1	<SHLI>	division by 2
4	16	\$0000.0400	<energy>	DIV2	1	<SHRI>	multiplication by 2
5	32	\$0000.0100: 4A00.8018	<DO> <en>	ILOG2	1	s.b. Func.3	mogarithm dualis
6	64	\$1000.C000: 0000.0000	<FP0> <en>	ISQRT	1	s.b. Func.4	square-root
7	128		s. 1.4.2	ICBRT	1	s.above 1.4.2	cube-root
8	256	\$0000.8000	<en-1> <en>	MOV	2	<MOV>	copying of one reg. before energy-reg.
9	512	\$0000.8000	<en-1> <en>	SWAP	2	s.b. Func.5	swap with reg. before energy-reg.
10	1024	\$0001.0000	<en-1> <en>	ADD	2	<ADD>	addition with "-"
11	2048	\$0002.0000	<en-1> <en>	SUB	2	<SUB>	subtraction of "-"
12	4096	\$0004.0000	<en-1> <en>	MUL	2	<MUL>	multiplied with "-"
13	8192	\$0008.0000	<en-1> <en>	DIV	2	<DIV>	division by "-"
...

Fig. 14b

Function:	OpCodes of: (machine-code compilation of these mnemonics, here a Motorola-Example)
Func.1	CMPI 127,<E>; JLE <+2>; MOVI #127,<E>; CMPI -128,<E>; JGE <+2>; MOVI #-128,<E>
Func.2	TST <E>; JGE <+3>; MOVI #-1,<E>; JMP <+5>; JGT <+3>; MOVI #0,<E>; JMP <+2>; MOVI #+1,<E>
Func.3	MOVI #31,DO; BTST DO,<E>; JEQ <+3>; DJMP DO,<-2>; ADDI #1,DO; MOVE DO,<E>
Func.4	FILD <E>; FSQRT; FIST <E>
Func.5	MOVE <E-1>,-(A7); MOVE <E>,<E-1>; MOVE (A7)+,<E>

Fig. 14c

09704803 110300

Valuation-Function-Table: [VFT - Table of the valuation-functions]

column: datatype value-range meaning:

Valuation Function ID (PK)	signed short	± 32767	identifier of the valuation-function (energyspecif. neg.)
Valuation_Function_Type	char(1)	'E' 'A'	'E' = energy-valuation, 'A' = valuation for reaching closeness to programming-aim, ... (maybe further)
Valuation Function Mode	boolean	0 1	0 = SQL-mode ; 1 = machine-code mode
Valuation_Function	varchar2(99)	≤ 99 Bytes	valuation-function for energy- or aim-attainment
execution counter	integer	$0 \cdot 2^{32} - 1$	number of uses of this valuation-function
used_Functions_BitCode	number(19)	$0 \dots 2^{64} - 1$	BitCodes of all subfunctions
Function_ID_Chain	varchar2(99)	≤ 99 Bytes	chain of subfunctions (one byte \triangleq one Function ID)
avg Func execution time	integer	$0 \cdot 2^{32} - 1$	average by val.-func.-execution needed clock-cycles
boundary value counter	integer	$0 \cdot 2^{32} - 1$	counter incremented if the result is -128 or $+127$
low value counter	integer	$0 \cdot 2^{32} - 1$	counter incremented if the result inside ± 16
Valuation_Function_value	signed byte	$-128 \dots 127$	valuability of the valuation-function = SAC.Self-Valuation Aim/Energy(Valuation Function, Values)

Fig. 15a Initial Entries for Energy-Valuation and Aim-Closeness-Valuation:

ID	Ty	M	Valuation Function
-1	'E'	0	$\text{MAX}[\text{MIN}[\text{SGN}(\text{EnergyReg}' - \text{EnergyReg}^\circ) \cdot \text{SQRT}(\text{EnergyReg}' - \text{EnergyReg}^\circ) - 32 \cdot \text{CLT}(\text{i}).\text{Register_changed_BitCode} \& (\text{! } 2^{\text{Energy Register ID}}) \}, +127], -128]$
0	'A'	0	$\text{MAX}[\text{MIN}[\text{MAX}[\text{MIN}[\text{CLT}(\text{i}).\text{Register_changed_BitCode} \& \text{ADT.all_dest_Register_BitCode} \} + 16 \cdot \text{CLT}(\text{i}).\text{Register_source_BitCode} \& \text{ADT.all_source_Register_BitCode} \} + 32 \cdot \text{ADT.aim_fulfilled_Flag_Function}(\text{Aim_ID}) - \text{CLT}(\text{i}).\text{Processor_Mode_changed} - \frac{1}{4} \cdot \text{CLT}(\text{i}).\text{cycles_of_execution} - (\text{CLT}(\text{i}).\text{active_inactive_ChkSum_corrupt}) - (\text{CLT}(\text{i}).\text{Exception_vect_changed} > 0) - (\text{CLT}(\text{i}).\text{Number_of_Exception} > 0) - \frac{1}{2}(\text{CLT}(\text{i}).\text{OpCode_length_or_jump} > 4 \text{ or } \leq 0)], +127], -128]$

ex#	used F. BitCode	Function ID chain	ex.T	bdy#	low#	F.Val
0	\$189.0040.983B	2,5; 2,15; 12; 4,22,3,11,35,40,1,32,12; 11 ; 39	0	0	0	0
0	\$EE9.0001.3AAA	3,11,42,35,1,16,12; 3,12,41,35,1,16,12,10; 43,1,32,10, 3,7,11; 3,16,1,5,13,11; 3,3,11; 3,5,11 3,5,5,10; 3,8,5,10; 3,9,1,0,37,11;3,9,1,5,38,11;39	0	0	0	0

Fig. 15b**Status of the Artificial Consciousness: [SAC - status-values of the AC-program (only 1 row)]**

column: datatype value-range meaning:

Programm StartDate	timestamp	datetime	date and time of the start of the AC-program
actual Processor Mode	integer	$0 \cdot 2^{32} - 1$	flags above CCR control-register-bits
actual CPT_index	byte	$1 \dots 255$	$\text{CBT}(\text{max}(\text{i}) = \text{actual CPT Nr}) = \text{actual CPT}$
CxT counter	short	$1 \dots 65535$	number of creations of the dynamic CxT-tables
Aims_total	short	$1 \dots 65535$	number of total programming-aims
Aims soluted	short	$0 \dots 65535$	number of solved programming-aims
actual Aim ID	short	$0 \dots 65535$	ID of the actual programming-aim
Aim_Valuation_Mode	boolean	0 1	mode of the programming-aim-specific valuation-function: 0 = SQL-mode ; 1 = machine-code mode
Aim_Valuation_FunctionID	signed short	$0 \dots 32767$	actual VFT.Valuation_Function_ID referring the closeness to the programming-aim
Aim_Self_Valuation_Func	varchar2 (400)	max.400 Chars.	PL/SQL-valuation-function referring the efficiency of the valuation-function
Energy_Valuation_Mode	boolean	0 1	mode of the energyspecific valuation-function 0 = SQL-mode ; 1 = machine-code mode
Energy_Valuation Func ID	signed short	$-1 \dots -32768$	actual VFT.Valuation_Function_ID for energy-valuation
Energy_Self_Valuation_Func	varchar2 (400)	max.400 Chars.	PL/SQL-valuation-function referring the efficiency of the energyspecific valuation-function
max Valuation Function	signed short	$0 \dots 32767$	highest ID of all valuation-functions in the VFT.
min Valuation Function	signed short	$-1 \dots -32768$	lowest ID of all valuation-functions in the VFT.

Fig. 16

09704803 110300

Energy-Learn-Table: [ELT - appraises energyspecific actions in dependence of the used initial-conditions]
 column: datatype value-range meaning:

Energy_action (PK)	number	0..2 ¹²⁸ -1	max.16 byte opcode-combination of the action which changed the energy-register.
IniConNr (PK)	signed byte	-31..30	number of the used initial condition
Energy before	integer	0..2 ³² -1	energy-register before execution
Energy after	integer	0..2 ³² -1	energy-register after execution
min Operations BitCode	number	0..2 ¹²⁸ -1	bitcode of the probably used opcodes.
max Operations BitCode	number	0..2 ¹²⁸ -1	bitcode of the possibly used opcodes.
Register changed BitCode	number	1..2 ¹²⁸ -1	bitcode of the by action changed registers.
Register source BitCode	number	1..2 ¹²⁸ -1	bitcode of the probable source-registers.
used cycles of execution	short	1..65535	needed clock cycles for the energyspecific action
Energy valuation	signed byte	-128..127	result of the actual <i>VFT.Energy valuation Function</i>
Valuation Function ID	signed short	-1..-32768	used energyspecific valuation-function

Fig. 17

Energy-Base-Table: [EBT - evaluation of the energyspecific actions]
 column: datatype value-range meaning:

Energy_action (PK)	number	0..2 ¹²⁸ -1	max.16 byte opcode-combination of the action which changed the energy-register.
Execution_counter	byte	0..255	number of the ELT-entries until now.
FatalError_counter	byte	0..255	number of the occurred fatal errors: fatal errors correlate the columns 3-7 of the above learn-table, except <i>Divide-Error</i> or <i>Overflow-Exc.</i>
low_Error_counter	byte	0..255	number of <i>Divide-Errors</i> or <i>Overflow-Exceptions</i>
avg Energy after	integer	0..2 ³² -1	average energy-value after the action
all_Reg_dest_BitCode	number	0..2 ¹²⁸ -1	\exists ELT.Register changed Bitcode \forall ELT(OpCode)
cut_Reg_dest_BitCode	number	0..2 ¹²⁸ -1	ζ ELT.Register changed Bitcode \forall ELT(OpCode)
all_Reg_source_BitCode	number	0..2 ¹²⁸ -1	\exists ELT.Register source Bitcode \forall ELT(OpCode)
cut_Reg_source_BitCode	number	0..2 ¹²⁸ -1	ζ ELT.Register source Bitcode \forall ELT(OpCode)
max_Operation_BitCode	number	0..2 ¹²⁸ -1	\exists ELT.max Operation BitCode \forall ELT(OpCode)
min_Operation_BitCode	number	0..2 ¹²⁸ -1	ζ ELT.min Operation BitCode \forall ELT(OpCode)
all_Operation_BitCode	number	0..2 ¹²⁸ -1	\exists ELT.min Operation BitCode \forall ELT(OpCode)
cut_Operation_BitCode	number	0..2 ¹²⁸ -1	ζ ELT.max Operation BitCode \forall ELT(OpCode)
max write value	integer	0..2 ³² -1	maximum of all energy-values after energy-action
min write value	integer	0..2 ³² -1	minimum of all energy-values after energy-action
avg write value	integer	0..2 ³² -1	average of all energy-values after energy-action
max write gradient	integer	0..2 ³² -1	maximum gradient of the changes energy-register
min write gradient	integer	0..2 ³² -1	minimum gradient of the changes energy-register ##
avg write gradient	integer	0..2 ³² -1	average gradient of the changes energy-register
equal value probability	signed byte	-128..127	probability of equal result of energyspecific action
avg Energy gradient	signed int	$\pm 2^31$	average value-gradient of this energyspecific action
equal Gradient probability	signed byte	-128..127	probability: gradient is constant
avg_cycles of execution	short	1..65535	average needed clock cycles for this action
avg Energy valuation	signed byte	-128..127	result of the actual <i>VFT.Energy valuation Function</i>
Valuation_Function_ID	signed short	-1..-32768	ID of the used energy-valuation-function

Fig. 18

3.2 Flowchart of the AC-Program:

3.2.1 CxT(i) value assignments:

ORT & CRT(i) value assignments:

ORT.Register ID dest := $\log_2(\text{Bit}(\text{OLT.Register changed Mask}), \text{of the regarded changing})$
ORT.Register ID source := Register ID(C°), if ORT.calculation code > 0, otherwise -1
ORT.value before change := value(Register ID dest), before opcode-execution
ORT.value after change := value(Register ID dest), after opcode-execution
ORT.gradient if signed := MAX[MIN[ORT.value after change - ORT.value before change, +127], -128]
ORT.gradient if unsigned := MAX[MIN[ORT.value after change - ORT.value before change, +127], -128]
ORT.Operation_BitCode := $1 \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [\text{NF} \&\& (V_1^\circ < 0) \mid \mid \text{ZF} \&\& (V_1^\circ = 0)]$ $+ 2 \cdot [(V_1' = -V_1^\circ) \&\& \forall (V' = V^\circ)] + 4 \cdot [(V_1' = \sim V_1^\circ) \&\& \forall (V' = V^\circ)] + 8 \cdot [(V_1' = 0\text{LB}) \&\& \forall (V' = V^\circ)]$ $+ 16 \cdot [(V_1' = V_1^\circ + 0\text{LB}) \&\& \forall (V' = V^\circ)] + 32 \cdot [(V_1' = V_1^\circ - 0\text{LB}) \&\& \forall (V' = V^\circ)] + 64 \cdot [(V_1' = V_1^\circ \cdot 0\text{LB}) \&\& \forall (V' = V^\circ)]$ $+ 128 \cdot [(V_1' = V_1^\circ / 0\text{LB}) \&\& \forall (V' = V^\circ)] + 256 \cdot [(V_1' = V_1^\circ \% 0\text{LB}) \&\& \forall (V' = V^\circ)] + 512 \cdot [(V_1' = V_1^\circ \cdot 2 \wedge 0\text{LB}) \&\& \forall (V' = V^\circ)]$ $+ 2^{10} \cdot [(V_1' = V_1^\circ / 2 \wedge 0\text{LB}) \&\& \forall (V' = V^\circ)] + 2^{11} \cdot [(V_1' = V_1^\circ \mid 0\text{LB}) \&\& \forall (V' = V^\circ)] + 2^{12} \cdot [(V_1' = V_1^\circ \& 0\text{LB}) \&\& \forall (V' = V^\circ)]$ $+ 2^{13} \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [(\text{ZF} = 1) \&\& (2 \wedge 0\text{LB} \mid \sim V^\circ) \mid \mid (\text{ZF} = 0) \&\& (2 \wedge 0\text{LB} \mid V_1^\circ)]$ $+ 2^{14} \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [\text{NF} \&\& (V_1^\circ < 0\text{LB}) \mid \mid \text{ZF} \&\& (V_1^\circ = 0\text{LB})] + 2^{15} \cdot [(V_1' = C_1^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{16} \cdot [(V_1' = V_1^\circ + C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{17} \cdot [(V_1' = V_1^\circ - C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{18} \cdot [(V_1' = V_1^\circ \cdot C_1^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{19} \cdot [(V_1' = V_1^\circ / C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{20} \cdot [(V_1' = V_1^\circ \% C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{21} \cdot [(V_1' = 2^{C^\circ} \cdot V_1^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{22} \cdot [(V_1' = V_1^\circ / 2^{C^\circ}) \&\& \forall (V' = V^\circ)] + 2^{23} \cdot [(V_1' = V_1^\circ \mid C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{24} \cdot [(V_1' = V_1^\circ \& C_1^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{25} \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [(\text{ZF} = 1) \&\& (2 \wedge C_1^\circ \mid \sim V_1^\circ) \mid \mid (\text{ZF} = 0) \&\& (2 \wedge C_1^\circ \mid V_1^\circ)]$ $+ 2^{26} \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [\text{NF} \&\& (V_1^\circ < C_1^\circ) \mid \mid \text{ZF} \&\& (V_1^\circ = C_1^\circ)]$ $+ 2^{27} \cdot [(IP' \leq IP^\circ) \mid \mid (IP' > IP^\circ + 4)] \&\& (\text{Flags}' = \text{Flags}^\circ) \&\& \forall (V' = V^\circ)$ $+ 2^{28} \cdot [(IP' \leq IP^\circ) \mid \mid (IP' > IP^\circ + 4)] \&\& (\text{Flags}' = \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& (\text{NF} \&\& !\text{VF} \mid !\text{NF} \&\& \text{VF}) + \dots \forall \text{Jcc}(\text{CCR})$ $+ 2^{40} \cdot [(IP' \leq IP^\circ) \mid \mid (IP' > IP^\circ + 4)] \&\& (V_1' = V_1^\circ - 1) \mid \mid (V_1' = -1)] \&\& (\text{Flags}' = \text{Flags}^\circ) \&\& \forall (V' = V^\circ)$ $+ 2^{41} \cdot [(IP' = IP^\circ \pm 0\text{LB}) \&\& (SP) = IP^\circ] \&\& (\text{Flags}' = \text{Flags}^\circ) \&\& \forall (V' = V^\circ)$ $+ 2^{42} \cdot [(IP' = -4(SP)) \&\& (\text{Flags}' = \text{Flags}^\circ) \&\& \forall (V' = V^\circ)] + 2^{43} \cdot [(V_1' \neq V_1^\circ) \&\& (! \text{other_Integer_Operation_BitCode})]$ $+ 2^{44} \cdot [(V_F' \neq V_F^\circ) \&\& (! \text{other_FloatingPoint_Operation_BitCode})] + 2^{45} \cdot [(\text{CCR-Flags}' = 0) \&\& \forall (V_F' = 0)]$ $+ 2^{46} \cdot [(V_1' = C_F^\circ) \&\& \forall (V' = V^\circ)] + 2^{47} \cdot [(V_F' = C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{48} \cdot [(V_F' = V_F^\circ + C_1^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{49} \cdot [(V_F' = V_F^\circ - C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{50} \cdot [(V_F' = V_F^\circ \cdot C_1^\circ) \&\& \forall (V' = V^\circ)] + 2^{51} \cdot [(V_F' = V_F^\circ / C_1^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{52} \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [\text{NF} \&\& (V_F^\circ < C_1^\circ) \mid \mid \text{ZF} \&\& (V_F^\circ = C_1^\circ)]$ $+ 2^{53} \cdot [(V_F' = 1.0) \mid \mid (V_F' = 0.0) \mid \mid (V_F' = \pi) \mid \mid (V_F' = e)] \&\& \forall (V' = V^\circ)$ $+ 2^{54} \cdot [(V_F' = -V_F^\circ) \&\& (V_F^\circ < 0) \&\& \forall (V' = V^\circ)] + 2^{55} \cdot [(V_F' = C_F^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{56} \cdot [(V_F' = V_F^\circ + C_F^\circ) \&\& \forall (V' = V^\circ)] + 2^{57} \cdot [(V_F' = V_F^\circ - C_F^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{58} \cdot [(V_F' = V_F^\circ \cdot C_F^\circ) \&\& \forall (V' = V^\circ)] + 2^{59} \cdot [(V_F' = V_F^\circ / V_F^\circ) \&\& \forall (V' = V^\circ)] + 2^{60} \cdot [(V_F' \cdot V_F' = V_F^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{61} \cdot [V_F' = \sin(V_F^\circ)] \&\& \forall (V' = V^\circ) + 2^{62} \cdot [V_F' = \cos(V_F^\circ)] \&\& \forall (V' = V^\circ) + 2^{63} \cdot [(V_F' = \text{atan}(V_F^\circ)) \&\& \forall (V' = V^\circ)]$ $+ 2^{64} \cdot [(V_F' = V_F^\circ \cdot 2 \wedge V_{F-1}^\circ) \&\& \forall (V' = V^\circ)] + 2^{65} \cdot [(V_F' = V_{F-1}^\circ \cdot \log_2(V_F^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{66} \cdot (\text{Flags}' \neq \text{Flags}^\circ) \&\& \forall (V' = V^\circ) \&\& [\text{NF} \&\& (V_F^\circ < C_F^\circ) \mid \mid \text{ZF} \&\& (V_F^\circ = C_F^\circ)] + 2^{67} \cdot [(V_1' = C_s^\circ) \&\& \forall (V' = V^\circ)]$ $+ 2^{68} \cdot [(V_s' = C_1^\circ) \&\& \forall (V' = V^\circ)] + \dots$, where $V' = \text{value_after_change}$ ($\neg \text{Flags}$), $V^\circ = \text{value_before_change}$ $C^\circ = \text{value}(\text{Register_ID_source})$. Here has to be checked over all Register_ID_source(eq.kind). Though equal Register-ID's in the PK several bits can be set. [p.e. because $4 = 2 + 2 = 2 \cdot 2 = \text{SHL}(2) = \dots$]

Fig.19

OLT & CLT(i) value assignments:

OLT.Processor_Mode_Changed := $\lambda \{ \text{EFlags}' / \text{SR}_\mu \& ! \exists 2 \wedge \text{CCR_Flags} \} > 0 \mid \mid$ ORT.value after change(Register ID of a special-register)
OLT.aim_valuation := VFT.Aim_Valuation_Function(SAC.Aim_Valuation_FunctionID, ORT.xxxxxx, Registers_changed_BitCode, Registers_source_BitCode, min_Operations_BitCode, max_Operations_BitCode, used_cycles_of_execution, ...)
CLT(n).gradient_aim_valuation := CLT(n).aim_valuation - CLT(n-1).aim_valuation
all other column-assignments are declared adequate in the OLT-description in fig.6.

Fig.20

09704803 " 110300

OBT & CBT(i) value-assingments:

OBT.Execution_counter := Execution_counter + 1
OBT.FatalError_counter := FatalError_counter + (0 < OLT.Number_of_Exception ≠ Divide_Error, Overflow) OLT.active_ChkSum_corrupt OLT.inactive_ChkSum_corrupt OLT.Exception_vect_changed OLT.Processor_Mode_changed)
OBT.Jump_longOp_probability := MAX[MIN[Jump_probability + (OLT.OpCode_length_or_jump ≤ 0) + (OLT.OpCode_length_or_jump > 4), +127], -128]
OBT.avg_OpCode_jump_length := (execution_counter * avg_OpCode_jump_length + akt.OpCode_jump_length) / (execution_counter + 1)
OBT.OpCode_len_unconfirmed := OpCode_len_unconfirmed (avg_OpCode_length ≠ act.OpCode_length)
OBT.avg_cycles_of_execution := (execution_counter * avg_cycles_of_execution + act.cycles_of_execution) / (execution_counter + 1)
OBT.exec_cycles_unconfirmed := exec_cycles_unconfirmed (avg_cycles_of_execution ≠ act.cycles_of_execution)
OBT.Register_write_probability := MAX[MIN[Register_write_probability + 2*[(min.Reg.ID ≤ ORT.Column_ID_OLT ≤ max.Reg.ID) && ORT.value_before_change ≠ ORT.value_after_change] - 1, +127], -128]
OBT.Register_copy_probability := MAX[MIN[MIN(Register_copy_probability + 2*[(min.Reg.ID ≤ ORT.Column_ID_OLT ≤ max.Reg.ID) && ORT.value_before_change ≠ ORT.value_after_change && (min.Reg.ID ≤ ORT.Column_ID_source ≤ max.Reg.ID)] - 1, +127], -128]
OBT.Memory_write_probability := MAX[MIN[Memory_write_probability + 2*[(min.Adr.Reg.ID ≤ ORT.Column_ID_OLT ≤ max.Adr.Reg.ID) && ORT.value_before_change ≠ ORT.value_after_change] - 1, +127], -128]
OBT.Memory_copy_probability := MAX[MIN[Memory_copy_probability + 2*[(min.Adr.Reg.ID ≤ ORT.Column_ID_OLT ≤ max.Adr.Reg.ID) && ORT.value_before_change ≠ ORT.value_after_change && (min.Adr.Reg.ID ≤ ORT.Column_ID_source ≤ max.Adr.Reg.ID)] - 1, +127], -128]
OBT.Reg_to_Mem_probability := MAX[MIN[Reg_to_Mem_probability + 2*[(min.Adr.Reg.ID ≤ ORT.Column_ID_OLT ≤ max.Adr.Reg.ID) && ORT.value_before_change ≠ ORT.value_after_change && (min.Reg.ID ≤ ORT.Column_ID_source ≤ max.Reg.ID)] - 1, +127], -128]
OBT.Mem_to_Reg_probability := MAX[MIN[Mem_to_Reg_probability + 2*[(min.Reg.ID ≤ ORT.Column_ID_OLT ≤ max.Reg.ID) && ORT.value_before_change ≠ ORT.value_after_change && (min.Adr.Reg.ID ≤ ORT.Column_ID_source ≤ max.Adr.Reg.ID)] - 1, +127], -128]
OBT.Multi_Reg_write_prob := <i>like in Register_write_probability, but with min.2 appropriate ORT.Column_ID_OLT -entries.</i>
OBT.Multi_Mem_write_prob := <i>like in Memory_write_probability, but with min.2 appropriate ORT.Column_ID_OLT -entries.</i>
OBT.Multi_Reg_to_Mem_prob := <i>like in Reg_to_Mem_probability, but with min.2 appropriate ORT.Column_ID_OLT + Column_ID_source -entries.</i>
OBT.Multi_Mem_to_Reg_prob := <i>like in Mem_to_Reg_probability, but with min.2 appropriate ORT.Column_ID_OLT + Column_ID_source -entries.</i>
OBT.xxx_Reg_source dest BitCode: <i>see table-description</i>
OBT.xxx_calculation BitCode: <i>see table-description</i>
OBT.max_write_value := MAX(max_write_value, ORT.value_after_change)
OBT.min_write_value := MIN(min_write_value, ORT.value_after_change)
OBT.avg_write_value := (execution_counter * avg_write_value + ORT.value_after_change) / (execution_counter + 1)
OBT.max_write_gradient := MAX(max_write_gradient, ORT.value_after_change - ORT.value_before_change)
OBT.min_write_gradient := MIN(min_write_gradient, ORT.value_after_change - ORT.value_before_change)
OBT.avg_write_gradient := (execution_counter * avg_write_gradient + ORT.value_after_change - ORT.value_before_change) / (execution_counter + 1)
OBT.evaluated_source_[Num]Register := <i>probability-function</i> (xxx_Reg_source_BitCode, confirmation_counter)

09704803-110300

3.2.4 AC-flowchart:

a.) Initial Preparations:

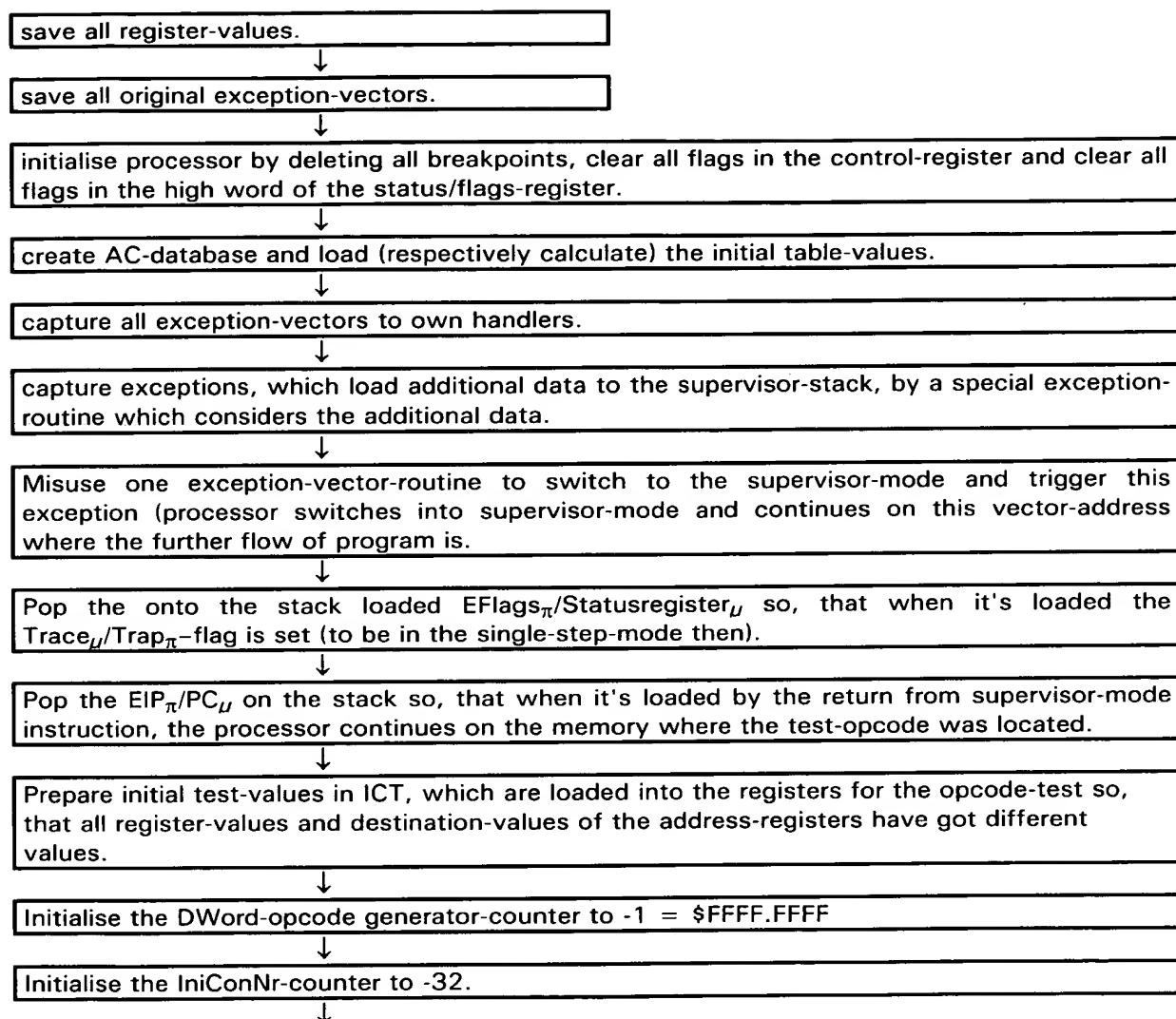


Fig.24a

09704803 110300

b.) Base-Learning:

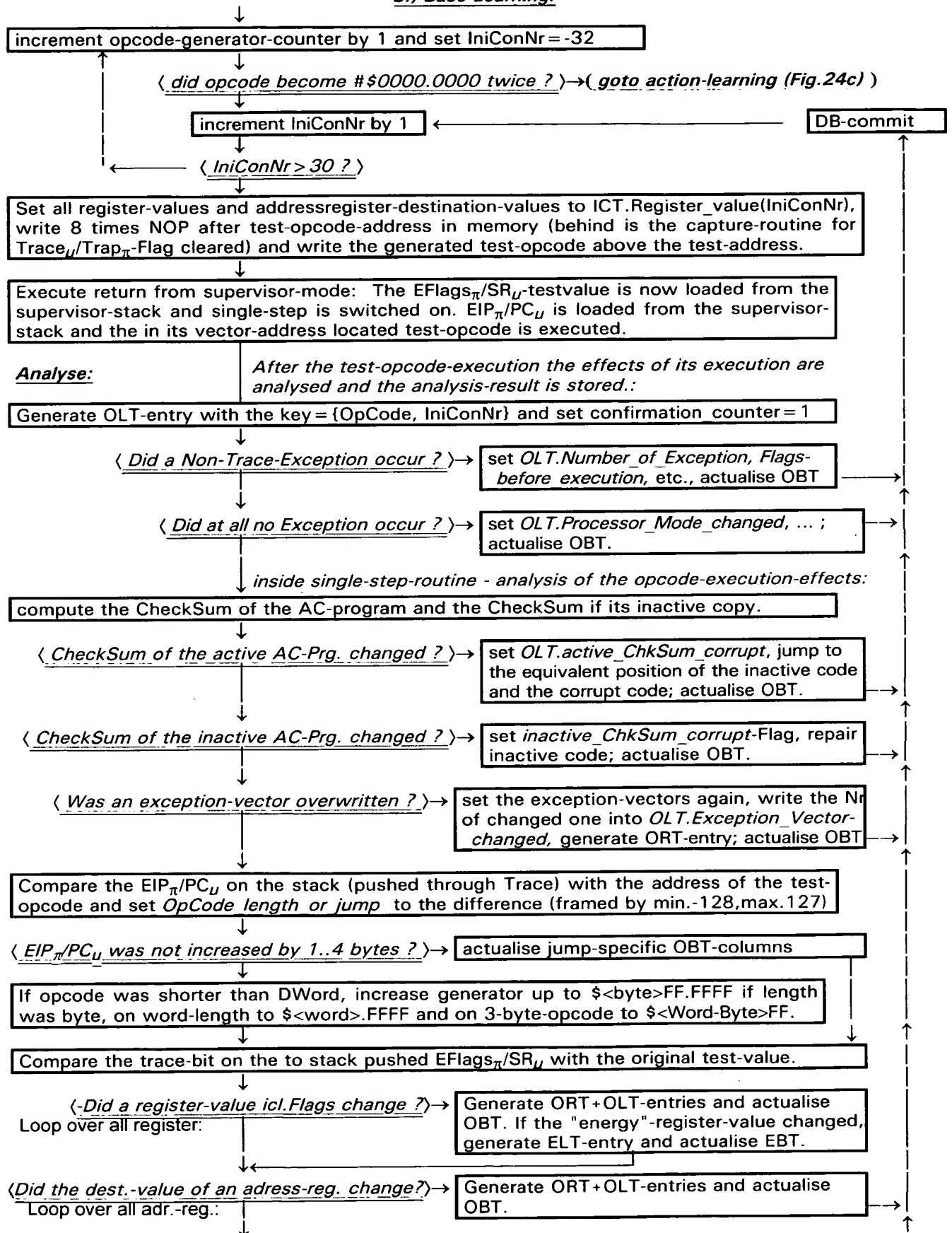


Fig.24b

00000000 00000000 00000000 00000000

c.) Double-OpCode-Acting:

(*Begin of Double-OpCode-Acting [after Base-Learning - Fig.24b]*)

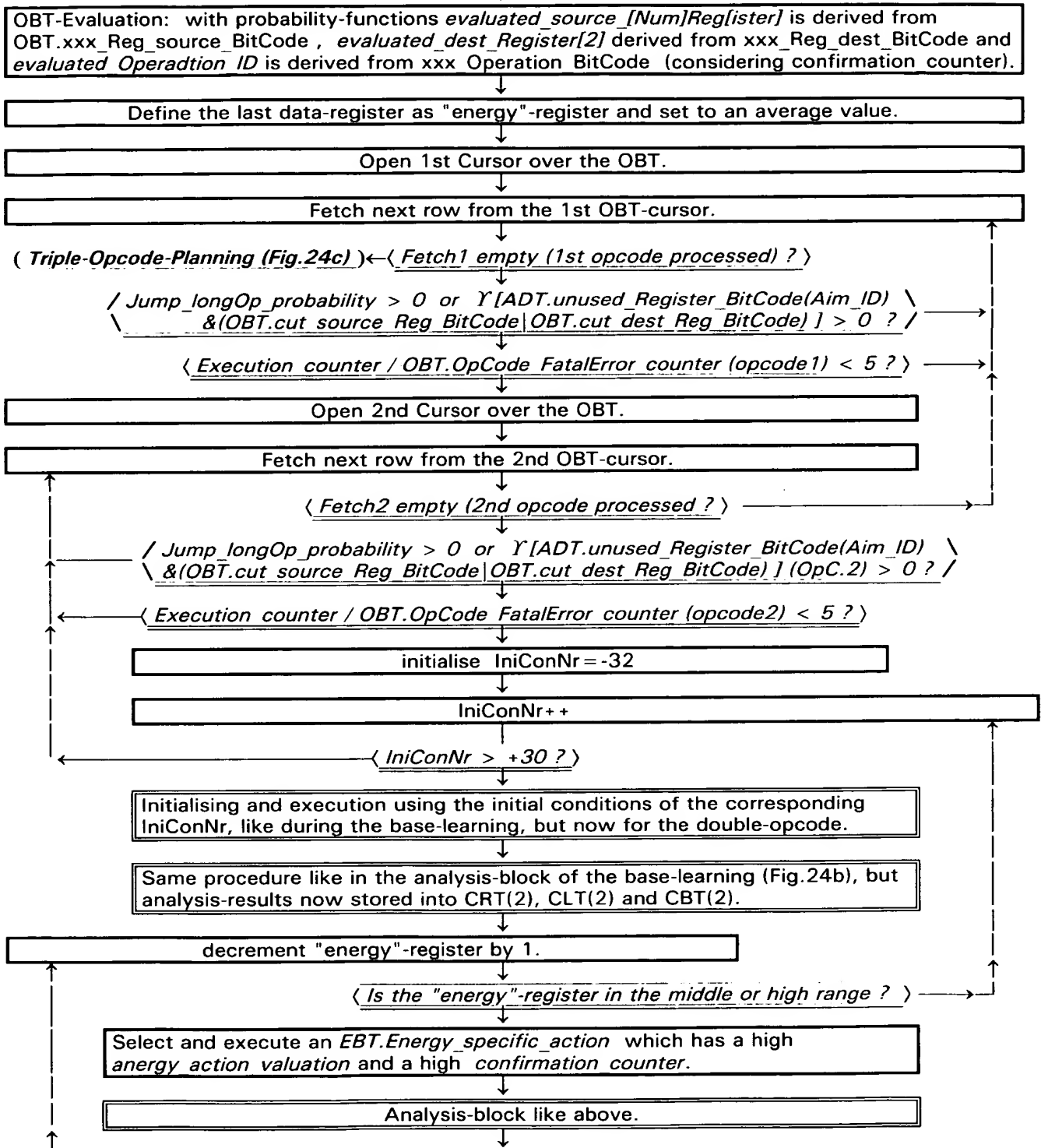


Fig.24c

09704803 "110300

d.) Triple-OpCode-Planning:

(*Begin of Triple-OpCode-Planning [after Double-OpCode-Actions]*)

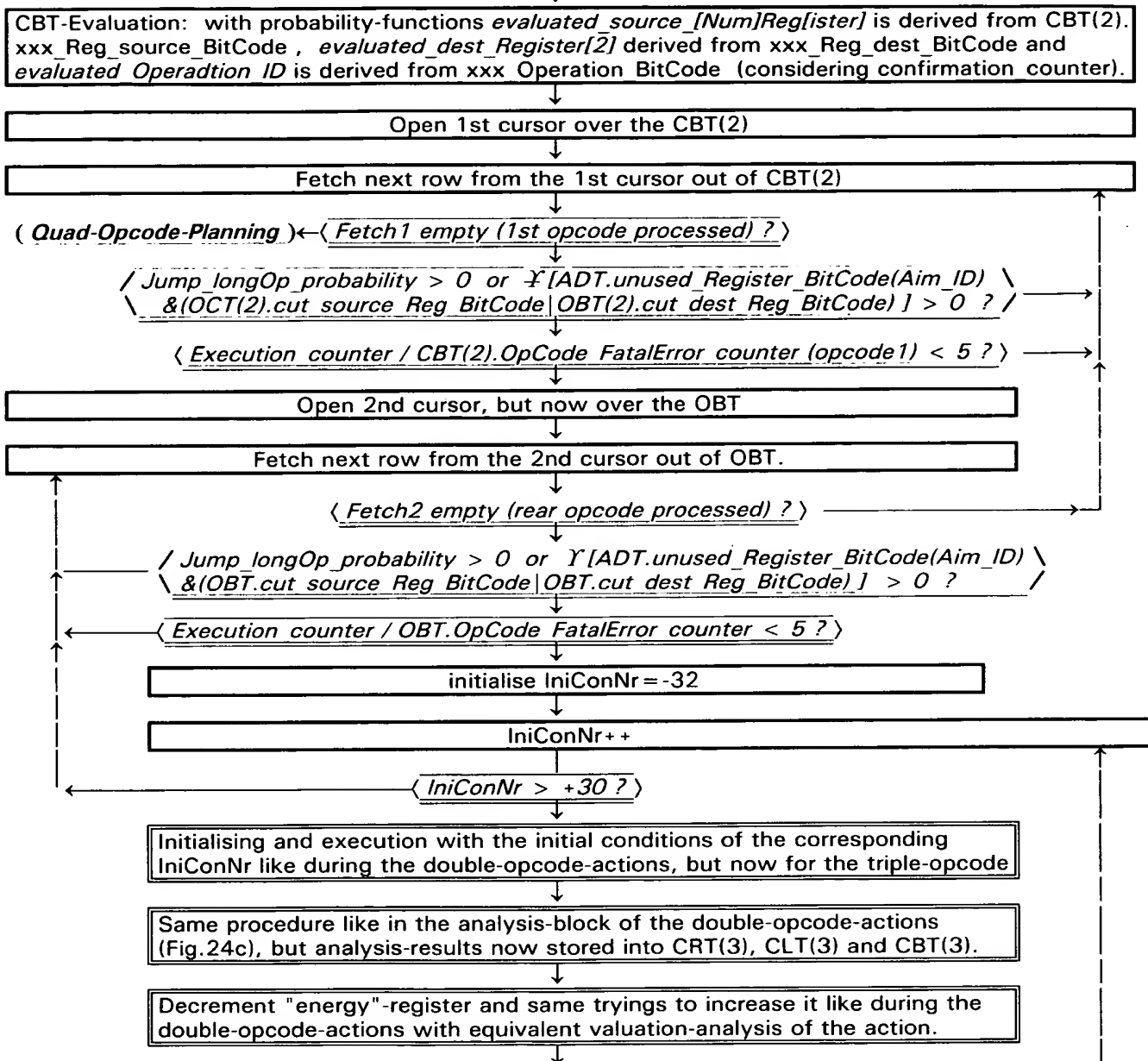


Fig.24d

Procedure for higher combinations analogous, using CxT(n), where n = sum_of_opcodes.

Gerd Kramm